Arageli: a Library for Algebraic Computations. Overview of Architecture

Sergey S. Lyalin and Nikolai Yu. Zolotykh

N.I. Lobachevsky State University, Gagarin ave. 23,
Nizhni Novgorod, 603950, Russia
sergey.lyalin@itlab.unn.ru, zolotykh@vmk.unn.ru

Abstract. Principles and design of Arageli library are reported. A short overview of abilities, algorithms, structures, and subsystems are given.

1 Introduction

Arageli [1] is a library for dealing with precise, i.e. symbolic or algebraic, computations. It contains a definition of basic algebraic structures such as integer numbers with arbitrary precision, rational numbers, vectors, matrices, polynomials etc.

The excellent solutions and the best development have been absorbed into the library. On the other hand, Arageli has such features that have been formed under the influence of historical circumstances and manners of developers.

The library was created in Pascal language originally. Further the developers chose C++. This gave a possibility to use parametric types and opportunity to construct types with all features of original algebraic structures in compile time.

Now Arageli is a library that uses all abilities of C++. Power and expressiveness of the language allow us to realize risky ideas and solutions that are absent in other libraries of the same level.

Arageli is used in several research projects concerning exact linear algebra, linear and integer linear programming, polyhedra and its integer points, integer factorization, symbolic computations. The library accumulates frequently used things from the projects.

Several researches contributed to Arageli at various times. Here is a list in alphabetical order: Eugene Agafonov, Max Alekseyev, Aleksey Bader, Sergey S. Lyalin, Aleksey Polovinkin, Andrey Somsikov, Nikolai Yu. Zolotykh.

This work is partially supported by Russian Foundation for Basic Research; Grant No. 05-01-00552-a.

2 Design Overview

The library supports the following structures (including basic operations on them): integer number with arbitrary precision, floating point number with arbitrary precision, rings of fractions (including rational numbers, rational functions

etc), univariate polynomials (dense and sparse representations), multivariate polynomials (dense and sparse representations), vectors, matrices, finite fields, residue rings (modular arithmetic), algebraic numbers, convex polyhedra.

Almost all structures listed above are represented as generic (templated) classes. User can construct any combinations of basic structures. This distinguishes Arageli from such library as NTL [5] and others which don't provide an opportunity to construct new types.

The user has an ability to use a reference counter with all aggregate types (i.e. such types that has several arbitrary number of other elements as a part). It decreases the number of objects copying. This parameter is controlled statically as a template parameter.

Arageli has its own implementation of arithmetic with arbitrary precision, but it is slow and was created for purposes of illustration. In real computations the user can use faster implementation of arithmetic: GMP C++ wrappers [3] mixing with other structures and algorithms of Arageli. Total parameterization allows doing that.

In respect to algorithms the library includes the following: Euclidean and extended Euclidean algorithm for GCD; Euclidean binary algorithm; Gaussian elimination and Gauss-Bareiss algorithm for the reduced row echelon form of a matrix for field and integer domain respectively; Smith diagonal form for integer and polynomial matrices; Solovay-Strassen, Miller-Rabin and Agrawal-Kayal-Saxena test for primality; integer number factorization by Pollard's ρ-method and Pollard's $p-1$ method; solving linear programming problem by simplex method, integer linear programming problem by Gomory's methods; location of roots with rational bounds by Sturm's algorithm; RSA encryption; double description method for polyhedral cone; triangulation of polyhedral cone; LLL algorithm for finding reduced basis of a lattice.

All listed algorithms are implemented as C++ templates. Moreover we use the special way to control an algorithm while it is working and put out intermediate results of the computations which is called *controller*. It is useful for demonstrative purposes and in education process. The controller is statically determined at the point of the algorithm instantiation because the controller is a template parameter. There are no overheads without additional management enabled.

Fully parameterized algorithm functions are useful when we need to calculate the number of specific operations under objects that are processed by the algorithm. We can replace original types by wrappers with possibility to calculate number of operations. It is good at simulation on presented algorithms with precise estimation of computation time because we abstract from machine architecture.

As a complex system Arageli has the modular and multi-level structure. Generic character of implemented things makes easier separating components that are conceptually independent. On the other hand, it makes possible simple mixing ones to single task tool. Each separated part of Arageli can be replaced by other similar one without any problems.

Arageli allows to perform mixed computations. All functions (including operators) that have more than one argument have ability to take different types for *similar arguments*. For example, operator+ for two vectors can perform elementwise addition on vectors with different types of elements. There are several rules (that automatically verifies by a compiler) that define more precisely the meaning of similarity of argument types and allowable type propagations. On the bottom layer it is an extension of the little modified rules of the mixed computations in C++ [6].

All algorithms and structures are exception safe. Information about each significant error in the library is represented as a C++ exception. There are several debugging levels of the Arageli execution. Almost all reasonable checks are verified on different levels.

There is complicated input/output subsystem. It allows performing input/output in reversible, human readable formats. Also it is good at aligned outputting to a monospaced console and LaTeX output of complex algebraic structures. In the future we are planning to implement some more formats to communicate with well-known mathematical packages the same way as in LiDIA [4].

The library is open. It means that the user can add own part to Arageli and use own types with Arageli structures and algorithms. There is a special mechanism to facilitate this work for the user. In many cases for all algorithms and structures inside Arageli it doesn't matter what the origin of a type used by user. It makes Arageli very flexible like, for example, Boost library [2].

Now Arageli is portable library. It can be compiled on several platform that have adequately implemented (according to the standard [6]) C++ compiler. We have tested Arageli on GCC, Intel C++ Compiler 8.0 and Microsoft C++ Compiler 7.1. Any non-standard features or architecture dependences (such as assembler inlines) are not used.

References

1. Arageli home page http://www.unn.ru/cs/arageli.
2. Boost home page http://www.boost.org.
3. GMP home page http://www.swox.com/gmp.
4. LiDIA home page http://www.informatik.tu-darmstadt.de/TI/LiDIA.
5. NTL home page http://www.shoup.net/ntl.
6. Programming languages — C++. International Standard ISO/IEC 14882:1998(E).